COMPSCI 311: Introduction to Algorithms **Spring 2020**

Homework 6

Released 4/16/2020 Due 4/29/2020 11:59pm in Gradescope

Instructions. You may work in groups, but you must write solutions yourself. List collaborators on your submission. Also list any sources of help (including online sources) other than the textbook and course staff. If you are asked to design an algorithm, please provide: (a) the pseudocode or precise description in words of the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.
Submissions. Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. Please assign pages to questions in Gradescope.

1. (20 points) **BoxDepth** (Erickson 12.2) The BoxDepth problem is as follows: Given a set of n rectangles in the plane, with sides parallel to the coordinate axes, what is the cardinality of the largest subset of these rectangles that contain a common point? Rectangles are given by bottom left and top right corners.

a) Describe a polynomial-time reduction from BoxDepth to MaxClique.

b) Describe and analyze a polynomial-time algorithm for BoxDepth. Extra credit for $O(n \log n)$ complexity.

c) Why don't these two results imply that P=NP?

2. (10 points) **DNF-SAT**. (Erickson 12.3) A boolean formula is in disjunctive normal form (or DNF) if it consists of a disjunction (OR) of several terms, each of which is the conjunction (AND) of one or more literals. DNF-Sat asks, given a boolean formula in disjunctive normal form, whether that formula is satisfiable.

a) Describe a polynomial-time algorithm to solve DNF-Sat.

b) What is the error in the following argument that P=NP?

Take an input to 3-Sat (a conjunction of clauses). Repeatedly apply distributivity to transform the formula to DNF. For example, $(x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y}) = (x \wedge \bar{x}) \vee (x \wedge \bar{y}) \vee (y \wedge \bar{x}) \vee (y \wedge \bar{y}) \vee (\bar{z} \wedge \bar{x}) \vee (\bar{z} \wedge \bar{y})$.
Then use the algorithm for DNF-Sat from (a) to determine in polynomial time whether the resulting formula is satisfiable. Since 3-Sat is NP-complete, we have P = NP!

3. (30 points) **Magic Boxes** (Erickson 12.5)

(a) Suppose a black box can determine in polynomial time, given an arbitrary weighted graph G, the length of the shortest Hamiltonian cycle in G. Describe and analyze a polynomial-time algorithm that computes, given an arbitrary weighted graph G, the shortest Hamiltonian cycle in G, using this black box as a subroutine.

(b) Suppose a black box can determine in polynomial time, given an arbitrary graph G, the number of nodes in the largest complete subgraph of G. Describe and analyze a polynomial-time algorithm that computes, given an arbitrary graph G, a complete subgraph of G of maximum size, using this black box as a subroutine.

4. (20 points) **Student Preferences** (Erickson 12.36) Jeff tries to make his students happy. He passes out a questionnaire that lists a number of possible course policies in areas where he is flexible. Every student is asked to respond to each possible course policy with one of "strongly favor", "mostly neutral", or "strongly oppose". Each student may respond with "strongly favor" or "strongly oppose" to at most five questions. Because Jeff's students are very understanding, each student is happy if (but only if) he or she prevails in at least one of their strong policy preferences. Either describe a polynomial-time algorithm for setting course policy to maximize the number of happy students, or show that the problem is NP-hard.

5. (20 points) **Concatenations** (K&T 8.23). Given a set of finite binary strings $S = \{s_1, \ldots, s_k\}$, we say that a string u is a concatenation over S if it is equal to $s_{i_1} s_{i_2} \ldots s_{i_t}$ for some indices $i_1, \ldots, i_t \in \{1, \ldots, k\}$. Consider the following problem: Given two sets of finite binary strings, $A = \{a_1, ..., a_m\}$ and $B = \{b_1, ..., b_n\}$, does there exist any string u so that u is both a concatenation over A and a concatenation over B?

A student claims: "At least the problem is in NP: I just have to show such a string u in order to prove the answer is yes." You point out that this explanation is insufficient: how do we know that the shortest such string is not exponential in the size of the input, in which case it would not be a polynomial-size certificate?

Fix the argument, proving that if there is a string u that is a concatenation over both A and B, then there is such a string whose length is bounded by a polynomial in the sum of the lengths of the strings in $A \cup B$.

6. **(0 points).** How long did it take you to complete this assignment?